Adabas & External Storage Access Functions

The external storage access functions enable an application program to read and write data to external storage devices. Five types of external storage access are available to Com-plete application programs:

- VSAM file I/O functions;
- ISAM and BDAM file I/O functions;
- SD functions;
- CAPTURE function;
- ADABAS interface.

Application programs can read and write data with VSAM data sets using VSAM access methods. In an MVS environment, application programs can use ISAM and BDAM data sets using Com-plete I/O functions. Note that data sets organized for access with an access method other than these cannot be accessed under Com-plete using standard Com-plete access functions.

SD functions enable the application program to read and write data either sequentially or directly to BDAM type data sets defined and allocated by the Com-plete SD access method. SD data sets are normally small volume data sets allocated within the Com-plete SD library. The SD functions of Com-plete enable the application program to allocate, read and write, and delete the data sets.

The CAPTURE function enables the application program to capture, or write, data to the Com-plete Capture data set. Records can be written by the application program, but cannot be read. Information written to the Com-plete Capture data set can be selected, or read, from the Capture data set using the Com-plete batch utility program CUCTCAPT.

The ADABAS interface enables your to access ADABAS files. ADABAS is the data base management system developed, marketed, and supported by SOFTWARE AG.

All external storage access functions can be used in the same application program, providing a wide range of data storage and access techniques.

This chapter covers the following topics:

VSAM File I/O

The Virtual Storage Access Method (VSAM) can be used by Com-plete application programs. Full use of any type of VSAM data set or I/O function is provided; however, LOCATE processing of VSAM data sets is not supported by Com-plete, because the VSAM buffers are maintained in a storage key different from that of the application program.

Com-plete performs all VSAM requests without using an exit list. At the completion of the request, the user's ACB exits are called if there are any errors. Asynchronous journalling requests are not supported.

The use of VSAM under Com-plete is transparent to you and the application program. With the exception of the use of terminal I/O functions, a batch VSAM program will execute as a Com-plete online program with no modifications.

COBOL support of VSAM requires the use of a COBOL compiler (COBOL/VS) that supports VSAM file processing. In general, all COBOL facilities provided for VSAM support can be used.

PL/I support of VSAM requires the use of a PL/I compiler that supports VSAM file processing. In general, all PL/I facilities provided for VSAM support can be used.

In order to implement a program that uses VSAM processing techniques in a Com-plete environment, you must fully understand the interface requirements. The basic interface consideration can be summarized as:

- VSAM file definitions to Com-plete;
- VSAM file definitions in programs;
- VSAM file OPEN;
- VSAM file I/O;
- VSAM file CLOSE.

Each of these considerations is discussed below.

File Definitions to Com-plete

Application programs refer to files using DD/DLBL names. To establish the link between these names and the corresponding data sets, all DD/DLBL names referenced by application programs must be declared ("cataloged" to Com-plete using the subfunction FM of online utility UUTIL. This declaration includes the data set name, disposition (MVS only), the name of the VSAM user catalog (VSE only), and other information (see below).

The data set is allocated to Com-plete dynamically when an OPEN request is issued against the appropriate DD/DLBL name and deallocated when Com-plete is stopped, or when the file is closed explicitly using the CLOSE or BATCH subfunctions of UUTIL FM.

In comparison with permanent data set allocation by JCL DD/DLBL statements, that was used with previous versions of Com-plete, this mechanism provides maximum flexibility of data set access by BATCH jobs and for data set maintenance (backup, restore, reallocation, rename, etc.) without the need of restarting Com-plete.

The file declaration in UUTIL FM not only contains the name of the data set, but also defines all parameters necessary for Com-plete to open the file. Options specified for a file in an application program are ignored. Instead, Com-plete uses the parameters defined via UUTIL FM to build control blocks and buffers once per DD/DLBL name. The parameters specified in the file definition to Com-plete must be consistent with the file processing techniques used by application programs.

All online programs referring to a given DD/DLBL name share the same control blocks and buffers. This allows efficient use of resources in the system, but also can significantly influence performance. Therefore, careful choice of parameter values is recommended for files referenced frequently and by a large number of terminal users.

The file definition parameters are described in the Com-plete Utilities documentation in the section Function FM - File (DDN) Catalog Maintenance.

File Definitions in Programs

File definitions in the application program should be created according to standard conventions established for the programming language being used.

From a Com-plete point of view, your only concern is that information placed in the application program must not conflict with the DDN catalog definition in the Com-plete program library.

File OPEN Statements

VSAM file OPEN statements in an online program are the same as those for a batch program. As in batch programs, the VSAM file must be opened via a standard OPEN statement before the first I/O request is made. In an online program, the OPEN statement is actually processed by Com-plete and not by MVS/VS VSAM. Com-plete establishes the necessary linkage to enable the application program to access the VSAM data set. After a VSAM file is opened by an online program, the file remains open to Com-plete until Com-plete is reinitialized or until the file is closed explicitly using the FM function of the UUTIL utility. Subsequent OPEN processing is bypassed, except for establishing the logical connection between the application program and Com-plete. Com-plete uses the DDN cataloged file definition to actually open the data set. Thereafter, all accesses to the file by an application program uses the existing ACB control block structure. This processing is transparent to the application program.

Note that a VSAM file actually remains open to Com-plete, not the application program, after the termination of the application program. VSAM files are physically closed only at Com-plete termination time or by a status change in the file initiated by use of the FM function in the UUTIL utility program.

If Com-plete encounters an error in preparing the file for access, an appropriate VSAM FILE STATUS value is returned to the application program. The FILE STATUS is returned by Com-plete’s OPEN processing or generated by Com-plete when indicating a non-VSAM error condition.

Non-VSAM FILE STATUS values that might be received from an OPEN request are:

136 x’88’	The file could not be opened because insufficient storage is available in the Com-plete region to build the necessary control blocks and buffers.Suggested response: Consult the system programmer responsible for Com-plete maintenance.
152 x’98’	Security errors. A password for the file was specified when the file was cataloged. This password was either not specified by the application program or, if provided, did not agree with the catalog entry. Suggested response: Provide the correct password or recatalog the DDN definition.
168 x’A8’	The DDNAME specified in the application program is invalid, not cataloged, or unavailable for online access because the file was placed in batch status. Suggested response: Ensure that the DDNAME specified in the application program is the same as that defined in UUTIL FM, and that online access to the file is enabled.

File I/O Operations

I/O operations to a VSAM file can proceed only if the file is successfully opened by the application program. The appropriate I/O request statements (START, READ, WRITE, DELETE, and REWRITE) can then be executed. Execution of these statements transfers control to Com-plete, and Com-plete performs the actual I/O.

Com-plete schedules the I/O request using VSAM, providing file integrity on a control interval level basis. If a record is requested that resides in a control interval being updated by another user, the request receives a VSAM FILE STATUS code of 96.

When Com-plete returns to the application program after processing an I/O request, the VSAM FILE STATUS value for the indicated file will have been set by Com-plete to either the value returned by VSAM or an appropriate value used to indicate a non-VSAM error detected by Com-plete.

Non-VSAM FILE STATUS values that might be received from I/O requests are:

24 x’18’	The file is no longer available. During processing of the application program and after the OPEN request has been processed, online access to the file was disallowed because another terminal user disabled access with the FM function in the UUTIL utility. Suggested response: Contact the system programmer responsible for Com-plete maintenance.
84 x’54’	LOCATE processing was requested. LOCATE processing is not supported by Com-plete.
153 x’99’	The type of processing requested (input or output) has been denied for the terminal user by the security system.
154 x’9A’	The type of processing requested (for example, update a record, add a record, read a record) conflicts with the options defined for the file in UUTIL FM.Suggested response: Correct the options defined in UUTIL FM.

File CLOSE Statements

When an online application program issues the CLOSE statement, Com-plete severs the logical connection between the program and the designated VSAM file(s). No subsequent I/O can be requested without requesting that the file(s) be reopened.

Com-plete automatically closes any VSAM file left open at termination of the application program; however, it is good practice and aids in system efficiency to close unused VSAM files. Note that the CLOSE statement causes a logical disconnection from the file and does not result in the actual close of the VSAM data set by Com-plete.

No FILE STATUS information is returned to the application program by Com-plete following a CLOSE operation.

ISAM & BDAM File I/O (MVS Only)

ISAM and BDAM files are accessed using the Com-plete ISAM and BDAM interface functions summarized in the following table.

Function	Description
TFDEQ	Release an exclusively held file to enable access by other programs.
TFENQ	Place an exclusive hold on a file to exclude access by other programs.
TFGET	Retrieve one or more records from a file.
TFGETU	Retrieve a record from a file, with the intention of updating that record.
TFPUT	Add one or more records to a file.
TFPUTU	Write an updated record to a file as an update to that file.

Before any of these functions can be used successfully, the following conditions must exist:

- Each file or data set to be accessed must be defined and allocated in the Com-plete initialization procedure;
- Each file or data set to be accessed must have a DDN entry defined in the Com-plete file catalog. This procedure, referred to as "cataloging the DDN," is fully described in the Com-plete Utilities documentation in the section *Function FM - File (DDN) Catalog Maintenance*.

Existing batch programs that access BDAM or ISAM files may continue to access those files with no changes to their logic. In addition, batch programs accessing BDAM and ISAM files should use standard file access techniques, rather than Com-plete file I/O functions.

For BDAM files, only RECFM=F is supported. If records are blocked when the file is created, Com-plete performs automatic deblocking when the file is accessed. BDAM files must be allocated with DCB=OPTCD=R (this can be overridden on the DD statement for the file in the Com-plete startup procedure). All BDAM modules should be placed in the pagefixed RAM/LPA list to avoid possible S606 abends.

For ISAM files, only fixed length records are supported. The maximum blocksize supported is 10K, and the relative key position must be larger than 0.

Retrieval of records for update is accomplished by using the TFGETU function. This function automatically forces an exclusive ENQ for the file containing the record to be updated. The exclusive ENQ is released when the TFPUTU function is executed to perform the actual record update. The ENQ/DEQ logic of the file I/O functions requires that the program be resident between execution of the two functions, TFGETU and TFPUTU; however, an ENQ remains in effect during a rollout caused by an ADABAS call. No other terminal I/O functions or rollout functions can be used between execution of the TFGETU and TFPUTU functions.

If the update of a record in a file is dependent upon information in one or more records in other files, the other files *must* also have an exclusive ENQ placed upon them for the duration of the update process. This is accomplished by using the TFENQ function. After the update process is complete, the TFDEQ function is used to release the exclusive ENQ. As indicated for the TFGETU and TFPUTU functions, the use of the TFENQ and TFDEQ functions requires the application program to be resident for the duration of execution of these functions. Therefore, the application program can not issue a terminal I/O function or a ROLLOUT function between execution of the TFENQ and TFDEQ functions. Note that an ENQ remains in effect during a rollout caused by an ADABAS call.

The use of multiple ENQs by more than one program should be considered carefully, since an interlock situation may occur. For example, if program A issues an ENQ for resource 1 and then resource 2, and program B issues an ENQ for resource 2 and then resource 1, a hard wait interlock situation will arise.

Request Parameter List

Use of the Com-plete file I/O functions requires definition of a working storage area within the application program called the Request Parameter List (RPL). The *RPL* contains information required by Com-plete to perform requested file I/O operations. A separate RPL can be defined for each file to be accessed, but only one RPL is Required.
The RPL may be shared by separate files, if the application program performs the necessary initialization functions; however, if more than one file is to be accessed simultaneously, separate RPL definitions must be established.

The format of the RPL is given in Request Parameter List .

TFDEQ Function (MVS Only)

The TFDEQ function is used to release an exclusive ENQ placed on a file against which the TFENQ function was executed. The name of the file to be released is specified in the RPL. All other information in the RPL is ignored.

An ENQ caused by execution of the TFENQ function is effective only while the executing program remains in storage. If the active program is rolled out of storage by Com-plete before the TFDEQ function is executed, the ENQ is lost prematurely, and the TFDEQ function has no effect. An ENQ remains in effect during a rollout caused by an ADABAS call, but it is cancelled automatically when a WRT or ROLOUT function is used. Therefore, if an application program issues a TFENQ function, neither terminal I/O functions nor the ROLOUT function can be executed until execution of the TFDEQ function is completed.

For example, if a program is performing a chained series of record additions to several files using data entered by a terminal operator, the program should read all the data from the terminal before issuing the TFENQ function against each file. After successfully adding the records to each file, the program should then issue the TFDEQ function for each file to release each exclusive ENQ.

Format

The format for using the TFDEQ function is:

```
TFDEQ (retcode,rpl)
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
rpl	Required.
	The name of the working storage area where the RPL is located.

The DDNAME field of the RPL must be initialized prior to issuing the TFDEQ function. All other RPL fields will be ignored.

Return Codes

The following return codes are issued by the TFDEQ function:

0	Normal completion.
4	A DEQ has already been issued for the file.

Abends

An abnormal termination may result during execution of the TFDEQ function. A possible cause is that the file specified by the DDNAME field of the RPL was not found.

TFENQ Function (MVS Only)

The TFENQ function is used to force an exclusive ENQ on a file in order to guarantee file integrity while record information is being accessed. The name of the file to be ENQed is specified in the RPL. All other information in the RPL will be ignored.

When a record in a file is to be updated, the TFGETU function is used to retrieve the record, and the TFPUTU function is used to update the record. The execution of the TFGETU function places an exclusive ENQ upon the file and thus ensures file integrity for the specified update operation. The ENQ is subsequently released with the TFPUTU function; however, if the record to be updated is dependent upon record information in one or more additional files, no ENQ will have been established for the additional files.

The TFENQ function is provided to enable the application program to force an exclusive ENQ on one or more interrelated files. After the required record(s) have been processed, any outstanding ENQ for such files must be released using the TFDEQ function.

An ENQ caused by execution of the TFENQ function is effective only while the executing program remains in storage. If the active program is rolled out of storage by Com-plete before the TFDEQ function is executed, the ENQ is lost prematurely and the TFDEQ function has no effect. An ENQ remains in effect during a rollout caused by an ADABAS call, but it is cancelled automatically when a WRT or ROLOUT function is used. Therefore, if an application program issues a TFENQ function, neither terminal I/O functions nor the ROLOUT function can be executed until execution of the TFDEQ function is completed.

For example, if a program is performing a chained series of record additions to several files using data entered by a terminal operator, the program should read all the data from the terminal before issuing the TFENQ function against each file. After successfully adding the records to each file, the program should then issue the TFDEQ function for each file to release each exclusive ENQ.

Format

The format for using the TFENQ function is:

```
TFENQ (retcode,rpl)
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
rpl	Required.
	Specifies the working storage item that contains the RPL for the indicated file.

The DDNAME field of the RPL must be initialized prior to issuing the TFENQ function. All other RPL fields will be ignored.

Return Codes

The following return codes are issued by the TFENQ function:

0	Normal completion.
4	An ENQ already exists for the indicated file, or the file is in batch status.
8	The application program has already ENQed the file.

Abends

An abnormal termination may occur during execution of the TFENQ function. A possible cause is that the file specified by the DDNAME field of the RPL was not found.

TFGET Function (MVS Only)

The TFGET function is used to retrieve one or more records from a file. The file to be accessed is identified through the use of an RPL.

The DDNAME field in the RPL identifies the file to be accessed. When the read operation is performed against the indicated file, the number of records to be read is determined by the NUMBER-RECORDS field of the RPL. The number of records specified is read and placed in the working storage area of the application program. The recipient area is identified as an argument in the TFGET function.

The type of file being accessed, either BDAM or ISAM, is not indicated in the application program, but is known to Com-plete through the DDname definition in UUTIL FM; however, the use of the TFGET function arguments is dependent upon the type of access method to be used. Note that an improper combination of arguments will result in abnormal program termination.

Three positional arguments are provided:

- RPL identifier
- Buffer area identifier
- Record identifier

Files, whether BDAM or ISAM, can be accessed sequentially or directly. The choice of access is identified in the RPL. If a file is accessed sequentially, the third argument of the TFGET function is ignored, regardless of the file organization type. When a file is accessed directly, the third argument must be provided.

If a BDAM file is to be accessed directly, the record argument must be provided. The record(s) to be retrieved can be identified by a relative record number (relative to 1) or by its actual disk address (MBBCCHHR), specified in the record argument. If more than one record is to be retrieved, the records retrieved will be consecutive records beginning with the relative record indicated. If actual addressing (MBBCCHHR) is used, only one block can be retrieved.

If an ISAM file is to be accessed directly, the record argument must be provided. The record to be retrieved is identified by the key provided in the record argument. If more than one record is to be retrieved, the records retrieved are consecutive records beginning with the record whose key is specified in the record argument.

Format

The format for using the TFGET function is:

```
TFGET (retcode,rpl,area[,record])
```

retcode	Required. A fullword where Com-plete places the return code upon completion of the operation.
rpl	Required. The request parameter list for the file.
area	Required. The buffer area in the program where the record(s) will be placed.
record	Optional. Default: None. This field is ignored if the file being accessed is processed sequentially.A variable length field containing the information required to perform a direct retrieval of a record. The information in this field is used by both BDAM and ISAM access methods; the format of the field depends upon which access method is to be used. For ISAM files, this field contains the key of the record to be retrieved.

Return Codes

The following return codes are issued by the TFGET function:

0	Normal return.
4	An exclusive ENQ is outstanding for the file, or the file is in batch status.
8	An I/O error occurred while accessing the file.
12	An end-of-file condition has occurred.
16	A "no record found" condition has occurred. This return code will be received only if the record was accessed directly.

Abends

An abnormal termination may occur during processing of the TFGET function. Possible causes include:

- The RPL is invalid;
- The file identified by the DDNAME field of the RPL was not found;
- There is not enough free space in the application program thread region to accommodate the necessary I/O buffer.

TFGETU Function (MVS Only)

The TFGETU function is used when updating a record in a file. When the TFGETU function is executed, an exclusive ENQ is directed against the file and the specified record is retrieved. The exclusive ENQ remains in effect until the record is rewritten using the TFPUTU function. No other program can access the file during this interval.

Note that an ENQ caused by execution of the TFGETU function is effective only while the executing program remains in storage. If the active program is rolled out of storage by Com-plete before the TFPUTU function is executed, the ENQ is lost prematurely. An ENQ remains in effect during a rollout caused by an ADABAS call, but it is cancelled automatically when a WRT or ROLOUT function is used. Therefore, if an application program issues a TFGETU function, neither terminal I/O functions nor the ROLOUT function can be executed until execution of the TFPUTU function is completed.

The DDNAME field in the RPL identifies the file to be accessed. When the read operation is performed against the indicated file, *only* one record is retrieved. The NUMBER-RECORDS field of the RPL is ignored. The record specified is read and placed in the working storage area of the application program. The recipient area is identified as an argument in the TFGETU function.

The type of file being accessed, either BDAM or ISAM, is not indicated in the application program; it is known to Com-plete through the DDname definition in UUTIL FM. Appropriate use of the TFGETU function arguments is dependent upon the type of access method to be used. An improper combination of arguments will result in abnormal program termination.

Three positional arguments are provided:

- RPL identifier
- Buffer area identifier
- Record identifier

Files, whether BDAM or ISAM, can be accessed sequentially or directly. The choice of access is identified in the RPL. If a file is accessed sequentially, the third argument of the TFGETU function is ignored, regardless of the file organization type. When a file is accessed directly, the third argument must be provided.

If a BDAM file is to be accessed directly, the record argument must be provided. The record(s) to be retrieved is identified by a relative record number (relative to 1) specified in the record argument.

If an ISAM file is to be accessed directly, the record argument must be provided. The record to be retrieved is identified by the key specified in the record argument.

Format

The format for using the TFGETU function is:

```
TFGETU (retcode,rpl,area[,record])
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
rpl	Required.
	The request parameter list for the file.
area	Required.
	The buffer area in the program where the record is to be placed.
record	Optional.
	Default: None. This field is ignored if the file being accessed is processed sequentially.A variable length field containing the information required to perform a direct retrieval of a record. The information in this field is used by both BDAM and ISAM access methods; the format of the field depends upon which access method is to be used. For ISAM files, this field contains the key of the record to be retrieved.

Return Codes

The following return codes are issued by the TFGETU function:

0	Normal return.
4	An exclusive ENQ is outstanding for the file, or the file is in batch status.
8	An I/O error occurred while accessing the file.
12	An end-of-file condition has occurred.
16	A "no record found" condition has occurred. This return code will be received only if the record was accessed directly.

Abends

An abnormal termination may occur during processing of the TFGETU function. Possible causes include:

- The RPL is invalid;
- The file identified by the DDNAME field of the RPL was not found;
- There is not enough free space in the application program thread region to accommodate the necessary I/O buffer.

TFPUT Function (MVS Only)

The TFPUT function is used to add one or more records to a file. The file to be accessed is identified through the use of an RPL.

The DDNAME field in the RPL identifies the file to be accessed. When the write operation is performed against the indicated file, the number of records to be added is determined by the NUMBER-RECORDS field of the RPL. The number of records specified is read from the working storage area of the application program and added to the file. The RPL fields NUMBER-OPTIONS and SEARCH-OPTIONS is ignored during execution of the TFPUT function.

The type of file being accessed, either BDAM or ISAM, is not indicated in the application program; it is known to Com-plete through the DDname definition in UUTIL FM. Appropriate use of the TFPUT function arguments is dependent upon the type of access method used. Note that an improper combination of arguments will result in abnormal program termination.

Three positional arguments are provided:

- RPL identifier
- Buffer area identifier
- Record identifier

If a BDAM file is being accessed, the third argument, record, must be provided. The record to be added is identified by a relative record number (relative to 1) specified in the record argument. If more than one record is to be added, the records are added in sequential order, beginning with the relative record indicated.

If an ISAM file is to be accessed, the third argument must be omitted. The record to be added is identified by the key specified in the record itself. If more than one record is to be added, the records are added consecutively, based on their keys.

Format

The format for using the TFPUT function is:

```
TFPUT (retcode,rpl,area [,record])
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
rpl	Required.
	The request parameter list for the file.
area	Required.
	The buffer area in the program where the record(s) to be added is located. The NUMBER-RECORDS field in the RPL indicates how many records exist in this area.
record	Optional.
	Used for BDAM files only.Default: None.A binary fullword containing the relative record number of the first record to be added. If multiple records are to be added, the relative record numbers of the records to be added begin with the one specified in the record argument and are incremented by one for each succeeding record. If actual addressing (MBBCCHHR) is used, only one record can be written.

Return Codes

The following return codes are issued by the TFPUT function:

0	Normal completion.
4	An exclusive ENQ is outstanding for the file, or the file is in batch status.
8	An I/O error occurred while accessing the file.
12	A duplicate record was found while adding an ISAM record.
16	An add for a record was requested, but there was not enough space available in the file.

Abends

An abnormal termination may occur while processing the TFPUT function. Possible causes include:

- The RPL structure was invalid;
- The file identified by the DDNAME field of the RPL was not found;
- There was not enough free space available in the application program thread region to allocate the required I/O buffer;
- A relative record number was not specified when adding a record to a BDAM file.

TFPUTU Function (MVS Only)

The TFPUTU function is used to rewrite one record to a file after it has been retrieved with the TFGETU function. The file to be accessed is identified through the use of an RPL.

The DDNAME field in the RPL identifies the file to be accessed. The record specified is read from the working storage area of the application program and rewritten to the file. The RPL fields NUMBER-RECORDS, NUMBER-OPTIONS and SEARCH-OPTIONS are ignored during execution of the TFPUTU function.

The type of file being accessed, either BDAM or ISAM, is not indicated in the application program; it is known to Com-plete through the DDname definition in UUTIL FM. Appropriate use of the TFPUTU function arguments is dependent upon the type of access method to be used. Note that an improper combination of arguments will result in abnormal program termination.

Two positional arguments are provided:

- RPL identifier;
- Buffer area identifier.

If a BDAM file is being accessed, the record to be rewritten is identified by the relative record number argument used in the last TFGETU function for the indicated file.

If an ISAM file is to be accessed, the record to be rewritten is identified by the key given in the record itself.

The TFGETU function is used to retrieve a record from a file when an update is to be performed to that record; however, the TFGETU function should always be preceded by the TFGET function. Any terminal I/O communication should be placed between the TFGET function and the TFGETU function. No terminal I/O must exist between the TFGETU function and the TFPUTU function.

When a record is retrieved, the information from that record is normally written to the terminal for display purposes with the intention of allowing the terminal operator to correct or modify the record for update purposes. The terminal write operation causes the application program to be rolled out of the thread, consequently nullifying any ENQ in effect for the designated file. Therefore, it is necessary to first retrieve a record with the TFGET function, perform the necessary terminal I/O communication, obtain the record a second time with the TFGETU function, verify that it has not changed, and update the record with the TFPUTU function.

Format

The format for using the TFPUTU function is:

```
TFPUTU (retcode,rpl,area)
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
rpl	Required.
	The request parameter list for the file.
area	Required.
	The buffer area in the program where the record(s) to be updated will be located.

Return Codes

The following return codes are issued by the TFPUTU function:

0	Normal completion.
8	An I/O error occurred while accessing the file.

Abends

An abnormal termination may occur while processing the TFPUTU function. Possible causes are:

- The RPL structure was invalid;
- An update was requested for a record not retrieved with a TFGETU function.

SD Files

SD files are direct access files, or data sets, that are allocated, accessed, and maintained using the SD access method of Com-plete. Each SD file allocated is suballocated within the Com-plete system SD library COM.SD.

The common name SD acknowledges that SD files can be accessed either sequentially or directly. The Com-plete SD access method is available to application programs through use of the SD functions. The SD functions are summarized in the following table. Each of these functions is described in detail in a later section in this chapter.

Function	Description
SDOPEN	Create an SD file.
SDWRT	Write a record to an SD file.
SDREAD	Read a record from an SD file.
SDCLOS	Close an SD file.
SDDEL	Delete an SD file from the disk.

SD files contain fixed length records and can be processed either randomly or sequentially. The maximum size of an SD fileâ€ƒdepends on the installation parameters of the Com-plete SD library. The maximum number of SD files that can concurrently be in use by an application program is five.

The SD file functions provide a flexible access method that allows many different kinds of access. Any specific SD file can be simultaneously accessed or updated from applications executing from different terminals at different COM-PASS levels and from batch applications.

SD files are used in the following ways:

- As work files unique to an application session, (for example, editor work space);
- As online data collection files to be processed by batch;
- As a User Profile mechanism for tailoring applications to the preferences of specific users;
- As data distribution techniques from batch programs to online applications.

SD files are uniquely identified by the combination of NAME and TID parameters specified in the SD file function call. All SD file function calls for a specific SD file should have identical NAME and TID operands.

Note that any sense of shared versus exclusive use of an SD file is totally the responsibility of the conventions that you establish.

You must choose values for NAME and TID that are consistent with the NAME and TID values used by other applications. That is, some applications require unique disk work areas, but other applications need to share a work area. The SD files can manage both.

Shared SD Files

Applications that share an SD file use the same fixed NAME and TID operands. For example, if the message-of-today needs to be displayed by all terminals accessing an application, then the application could specify a NAME of "TODAY" and a TID of "SHR" in each SD file call. All users of the application would be using one SD file.

Note that for applications that write shared SD files, no SD read-for-update/update or write-next-unwritten record facility exists. If multiple programs are adding records to an SD file, it is advisable to keep the record number of the next free record in the first record of the SD file. Each application would then:

1. ENQ/LOCK on a serialization resource.
2. Read the first record.
3. Note the next free record number.
4. Increment the next free record number.
5. Write back the first record.
6. EQ/UNLOCK the serialization resource.

7. Write the noted record.

Unique SD Files

Applications that require a unique SD file by user must use a technique that guarantees a unique NAME and TID combination.

Usually the terminal user’s user ID can be used as the NAME with a TID of "SHR" to ensure that an SD file is unique to a given user and that the SD file can be used by that user from any terminal.

If the SD file has meaning only during a single session with the application, then NAME could be specified as "APPL09" with the TID operand omitted. The TID could also be specified as the terminal ID, as retrieved by a GETCHR call. This allows the application to use the *hirec* operand on with SDOPEN or the length operand with SDREAD or SDWRIT.

A file name conflict may arise if a program that is simultaneously active on more than one level in a COM-PASS environment attempts to open an SD file. Com-plete provides a method to ensure that files (for example, work files) have unique names. The system does this by inserting the level number (a digit between 1 through 9) into the file name at user-specified positions. The required positions are denoted by high-value bytes (X‘FF’). Note that the first character position of the file name cannot be modified in this way.

In addition, if the TID value specified is negative (that is, the high order bit is set), Com-plete will interpret this as a request for a level-dependent SD-file. The user can specify the level required in the bottom 4 bits if these bits are all set (that is, X’0F’), then Com-plete will use the current level.

An SD file can be read, written, or deleted simultaneously by two or more application programs. If two or more application programs are processing an SD file and one of them requests a deletion of the file, the file is physically deleted only after the last program accessing the file has issued a close or delete request for the file.

SDOPEN Function

The SDOPEN function is used to create a new SD file or prepare an existing SD file for access. The SDOPEN function must be used by an application program prior to accessing or deleting an SD file. More than one SD file can be opened simultaneously, as long as five or less SD files are being accessed by the application program. If more than five separate SD files are to be accessed, then at least one SD file currently open will have to be closed before issuing another SDOPEN function.

When the SDOPEN function is issued, the status of the SD file is indicated by the return code value. If the SD file is new, a return code of 0 is given. If the SD file already exists, a return code of 4 is given.

If a new SD file is being opened, the SD function arguments that must be specified are:

- File name;
- Record length;

- Number of records.

If an existing SD file is being opened, the file name argument must also be specified; however, the arguments that specify record length and number of records is ignored by Com-plete and used to return the existing record length and the existing number of records to the application program. Note that the record length and number of records cannot be changed for an existing file.

When an existing SD file is opened, the highest record written in the SD file can be identified with the SDOPEN function by having Com-plete return the record number of the highest record written. This feature can be used by an application program if it has a need to determine the last record written in the SD file at any one time. This feature can also be used to determine the last record to read when reading an entire SD file sequentially, thus avoiding reading records that have not been written.

If a program attempts to open an existing SD file, and Com-plete determines that the SD file does not exist, Com-plete assumes that the open request is for a *new* SD file; therefore, when opening an existing SD file, valid argument values should always be specified for record length and number of records. If, after checking the return code, the application program determines that a new SD file has been opened instead of an existing SD file, the SD file can be deleted and an error message written to the terminal.

Format

The format for using the SDOPEN function is:

```
SDOPEN (retcode,filnam[,length][,numrec][,tid] [,hirec])
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
filnam	Required.
	An eight-byte alphanumeric field that contains the name of the SD file being opened.
length	Optional.
	Required for a new SD file.Default: For an existing SD file, the record length will be returned. The maximum record length is 32752.For a new SD file, a binary halfword containing the length of each record in the SD file.
numrec	Optional.
	Required for a new SD file.Default: For an existing SD file, the number of records allocated to the SD file is returned.For a new SD file, a binary halfword containing the number of records to be allocated for the SD file. The numrec for a new SD file remains as initialized in the application program. Since SD files are allocated on a track capacity basis, the application program can issue an SDCLOS function followed by an SDOPEN function for the SD file, and numrec will then contain the maximum number of records allocated for the SD file.
tid	Optional.
	Default: If the tid argument is omitted, the TID value for the terminal in use is assumed. The tid argument must be coded for an SD file being accessed by a batch program. For a new SD file, a binary halfword or a three-byte alphanumeric field containing the character string SHR. For an existing SD file, tid specifies a binary halfword. If the SD file was created with a tid value of SHR, then SHR must be specified.
hirec	Optional.
	Default: hirec is initialized to zeros by Com-plete for a new SD file. Com-plete returns the number of the highest record written for an existing SD file.For an existing SD file, a binary halfword in which Com-plete returns the record number of the highest record written to the SD file.

Return Codes

The following return codes are issued by the SDOPEN function:

0	A new SD file has been opened.
4	An existing SD file has been opened.
8	An existing SD file has been opened, but the SD file is currently being used by another application program.
12	The SD file has already been opened by the application program.

Abends

An abnormal termination may occur during execution of the SDOPEN function. Possible causes include:

- The SD file being opened is larger than the maximum size allowed;
- The SD file directory is full;
- An unrecoverable disk error has occurred.

SDWRT Function

The SDWRT function is used when writing records to an SD file. All records written are fixed-length records and can be written either sequentially or randomly.

The record to be written must reside in a working storage area of the application program. The application program can optionally write the entire record or a portion thereof.

When an SD file is created, each record is assigned a sequential number from one to the number of records requested. To write a record to an SD file randomly, the number of the record to be written must be specified. If no record number is specified, Com-plete writes the next sequential record to the SD file. Note that if this is the first SDWRT to a new SD file,*numrec*must be specified.

Note also that the length of each record to be written can be specified, but cannot be larger than the record length for the SD file as established when the file was created. If the length of data to be written is less than the record size for the file, the remaining characters of the record are padded with binary zeros. If the length of the record to be written is not specified in the SDWRT function, the record length established for the SD file during the SDOPEN function for the SD file is used.

Format

The format for using the SDWRT function is:

```
SDWRT (retcode,filnam,area[,numrec][,tid][,length])
```

retcode	A fullword where Com-plete places the return code upon completion of the operation.
filnam	Required.
	An eight-byte alphanumeric field that contains the name of the SD file to which the record is being written. The SD file specified by filnam must have been previously processed by the SDOPEN function.
area	Required.
	A buffer area in the working storage area of the application program that contains the record to be written.
numrec	Optional.
	Default: If numrec is not specified, the next sequential record is written. Required on the first SDWRT to a new SD file. A binary halfword that contains the number of the record to be written.
tid	Optional.
	Default: if tid is omitted, the TID value for the terminal in conversation is assumed. A tid must be specified for all batch programs.A binary halfword or a three-byte alphanumeric field that contains the character string SHR. The value must be the same as that specified when the file was opened.
length	Optional.
	Default: If not specified, the length is assumed to be the length specified in the SDOPEN function for the SD file.A binary halfword that contains the length of the data to be written.

Return Codes

A return code of 0 is issued upon normal completion of the SDWRT function.

Abends

An abnormal termination may occur during execution of the SDWRT function. Possible causes include:

- The SD file was not previously opened;
- The record number specified was greater than the maximum record number in the SD file;
- The record number specified was negative;
- An unrecoverable disk input/output error occurred.

SDREAD Function

The SDREAD function is used when reading records from an SD file. All records read are fixed-length records and can be read either sequentially or randomly.

The record to be read is placed in a working storage area of the application program. The application program can optionally read the entire record or a portion thereof.

When an SD file is created, each record is assigned a sequential number from one to the number of records requested. To read a record from an SD file randomly, the number of the record to be read must be specified. If no record number is specified, Com-plete reads the next sequential record from the SD file. If no records have been read yet, the first record is read.

Note that the length of each record to be read can be specified, but cannot be larger than the record length for the SD file as established when the file was created. If the length of data to be read is less than the record size for the SD file, the amount of data requested is read. If the length of the record to be read is not specified in the SDREAD function, the record length established for the SD file during the SDOPEN function for the SD file is used.

Format

The format for using the SDREAD function is:

```
SDREAD (retcode,filnam,area[,numrec][,tid][,length])
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
filnam	Required.
	An eight-byte alphanumeric field that contains the name of the SD file from which the record is being read. The SD file specified by filnam must have been previously processed by the SDOPEN function.
area	Required.
	A buffer area in the working storage area of the application program that contains the record after execution of the SDREAD function.
numrec	Optional.
	Default: if numrec is not specified, the next sequential record is read. If no records have been read, the first record is read. A binary halfword that contains the number of the record to be read.
tid	Optional.
	Default: If tid is omitted, the TID value for the terminal in conversation is assumed. A tid must be specified for all batch programs. A binary halfword or a three-byte alphanumeric field that contains the character string SHR. The tid value must be the same as that specified when the file was opened.
length	Optional.
	Default: If not specified, the length is assumed to be the length specified in the SDOPEN function for the SD file. A binary halfword that contains the length of the data to be read.

Return Codes

The following return codes are issued by the SDREAD function:

0	Normal completion.
4	The record requested has not been written to the SD file.
8	The record number requested is one larger than the last physical record in the SD file; this is the equivalent of an end-of-file condition.

Abends

An abnormal termination may occur during execution of the SDREAD function. Possible causes include:

- The SD file was not previously opened;
- The record number specified was more than one larger than the maximum record number in the SD file;
- The record number specified was negative;
- An unrecoverable disk input/output error occurred.

SDCLOS Function

The SDCLOS function is used to logically close an SD file. Because an application program can simultaneously process a maximum of only five SD files, the SDCLOS function must first be used if it is necessary to access more than five SD files.

Note that it is not necessary for an application program to issue an SDCLOS function. Com-plete closes the SD file when the application program terminates; however, it is recommended that SD files be closed when processing is completed. This facilitates more efficient operation of the system. In addition, since the SDOPEN function passes a return code value that indicates usage of an SD file by more than one application program, concurrent access of an SD file by more than one application program is logically communicated in proper access sequence.

Format

The format for using the SDCLOS function is:

```
SDCLOS (retcode,filnam[,tid])
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
filnam	Required.
	An eight-byte alphanumeric field containing the name of the SD file to be closed.
tid	Optional.
	Default: if not specified, the Terminal Identification number (TID) of the terminal in use is assumed. A tid must be specified by all batch programs that issue the SDCLOS function. A binary halfword or a three-byte alphanumeric field that contains the character string SHR.The tid must specify the same value as that specified when the SD file was opened.

Return Codes

The following return codes are issued by the SDCLOS function:

0	Normal completion.
4	SD still in use by another user.

Abends

An abnormal termination may occur during execution of the SDCLOS function. Possible causes include:

- The SD file was not opened;
- The *tid* argument was not specified in the batch program.

SDDEL Function

The SDDEL function is used to delete, or scratch, specific SD files. SD files to be deleted with the SDDEL function must be open at the time the SDDEL function is executed. Specifically, the SDCLOS function must not have been executed prior to executing the SDDEL function.

The SDDEL function identifies the SD file to be deleted by file name and TID. If an SD file was created with a TID value of SHR, then SHR must also be indicated in the SDDEL function.

If more than one application program is concurrently accessing an SD file and one of the accessing programs issues an SDDEL function for the SD file, the SD file is marked for deletion. However, the actual deletion occurs only after all application programs currently accessing the SD file have logically closed the SD file, either expressly by executing the SDCLOS function or implicitly by program termination.

Format

The format for using the SDDEL function is:

```
SDDEL (retcode,filnam[,tid])
```

retcode	Required.
	A fullword where Com-plete places the return code upon completion of the operation.
filnam	Required.
	An eight-byte alphanumeric field containing the name of the SD file to be closed.
tid	Optional.
	Required for batch programs.
	Default: If not specified, the Terminal Identification number (TID) of the terminal in use is assumed. A binary halfword or a three-byte alphanumeric field that contains the character string SHR. If specified, the tid must have the same value specified as that when the SD file was opened.

Return Codes

The following return codes are issued by the SDDEL function:

0	Normal completion.
4	The SD file specified is currently being used by another application program; deletion is pending.

Abends

An abnormal termination may occur during execution of the SDDEL function. Possible causes include:

- The SD file was not opened;
- The *tid* argument was not specified by the batch program requesting the SDDEL function.

CAPTUR Function

The CAPTUR function is used to write data from a program area to the Com-plete capture data set. The data is written from an area specified by an area argument for the length specified by a length argument. An identification argument is provided that must be used to pass a six-character identification code to the CAPTUR function, if the program issuing the CAPTUR function is a batch program. The identification argument is optional in an online program.

When a record is written to the capture data set, Com-plete automatically adds a variable-length header that identifies the record being written. The format of this header is shown in Captur Record Header.

The CAPTUR function can be used to:

- Keep a record of information for use in reports;
- Store add or update transactions in an online environment and then perform the actual file add or update I/O operation in batch at a later time;
- Record audit information;
- Record diagnostic messages created by an application program. The messages can then be used to trace execution of a program and assist in finding errors.

Format

The format for using the CAPTUR function is:

```
CAPTUR (retcode,area,length[,identifier])
```

retcode	Required. A fullword where Com-plete places the return code upon completion of the operation.
area	Required. The buffer area within the application program that contains the data to be captured.
length	Required. A binary halfword containing the length of the user data to be captured.
identifier	Optional. Default: If an identifier is not specified for an online program, the compressed name of the program issuing the CAPTUR function is placed in position zero of the CAPTUR record header. Required for batch programs. An alphanumeric field containing a six-byte identifier to be placed in position 14 of the CAPTUR record header. An identifier is required if the program issuing the CAPTUR function is a batch program, but is optional for online programs.

Return Codes

The following return codes are issued by the CAPTUR function:

0	CAPTUR was successful.
4	CAPTUR was not successful. This condition would arise if the CAPTUR feature were disabled.

Abends

Abnormal termination of the application program issuing the CAPTUR function can occur when the CAPTUR function itself abnormally terminates. Possible causes for this condition are:

- The *length* argument specified was invalid;
- An argument address is invalid;
- The identifier argument was omitted for a batch program.

ADABAS Interface

Application programs that communicate with ADABAS must use the standard ADABAS calling sequence interface as illustrated below.

```
ADABAS (argument1,...,argument6)
```

The arguments available with the ADABAS call statement are fully described in the *ADABAS Command Reference documentation*.

Note that this call differs from other calls in that the RETCODE parameter must not be specified.

Multiple ADABAS Nuclei

Access to multiple ADABAS nuclei is accomplished by setting the file number in the ADABAS Control Block (ACB).

The ACB file number is a binary halfword. Set the file number to the actual file number plus 256 times the data base ID.

If the data base ID is not specified, the default ID set at Com-plete initialization or as set by the installation’s ULOPADAB exit.

Note:
Any ACB file number specified is cleared on return from ADABAS.

The Com-plete/ADABAS interface passes the supplied parameter list to a standard ADALNK module. Be aware that the ADALNK (and ULOPADAB) are entered in the A-mode of the calling routine: this means that if the program is running 31-bit mode, the parameter list must contain valid 31-bit addresses.

Return Codes Abends

The response to the ADABAS call is returned in the ADABAS control block. Please see the ADABAS documentation for further details. Abnormal termination of the application program issuing the ADABAS call may occur if the parameter list contains invalid addresses.